Step 1: Let $V_1 := \{1\}$ and T_1 the subtree of T consisting only of node 1. Alignment $\mathcal{A}^{(1)}$ then is equal to S_1.

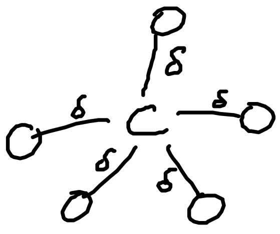
Step 2: For i from 1 to $k-1$ repeat the following steps:
(a) Choose any node $s \notin V_i$, neighbored to a node $r \in V_i$ in T. Let $V_{i+1} := V_i \cup \{s\}$ and add s and edge $\{r, s\}$ to T_i. $\Rightarrow T_{i+1}$.
(b) Compute optimal alignment $\mathcal{A}$ of S_s and row $\mathcal{A}^{(i)}_r$ of alignment $\mathcal{A}^{(i)}$ corresponding to S_r. As $\delta(-,-) = 0$ holds, the score of an optimal alignment of S_s and S_r is equal to the score of an optimal alignment for S_s and $\mathcal{A}^{(i)}_r$.
(c) For each gap added to $\mathcal{A}^{(i)}_r$ by $\mathcal{A}$ add a *gap column* to $\mathcal{A}^{(i)}$. Finally add the row of $\mathcal{A}$ corresponding to S_s to the modified $\mathcal{A}^{(i)}$ creating $\mathcal{A}^{(i+1)}$.

Step 3: Output alignment $\mathcal{A}^{(k)}$.

Observations:

- The algorithm obviously guarantees consistency for the new edge $\{r, s\}$ and adding further edges doesn't change consistency of previously added edges as only gap columns are added.
- As a tree with k nodes always has $k-1$ edges $\Rightarrow$ the loop in step 2 will process each edge of the alignment tree eventually. $\square$

Special case: Alignment tree for $\mathcal{S} = \{S_1, \dots, S_k\}$ is a star (i.e. a tree with center c and $k-1$ leaves, each connected to c by an edge).
This special case is often called *star alignment*. The algorithm for the star alignment first determines the center c from the given words S_i as follows:



- For each $1 \leq i \leq j \leq k$ determine similarity $sim(S_i, S_j)$ of S_i and S_j according to the scoring function used.
- Choose c to be the word w with minimal sum $\sum_{S \in \mathcal{S}} sim(w, S)$.

Then T is the star with center c, leaves $S \setminus \{c\}$ and the previously determined similaryties as edge labels. Afterwards the algorithm from the previous proof is used to create a multiple alignment consistent with T. This method is called *center star method*.

It should be obvious that this method does not create an optimal alignment for the input. The question is how good the created alignment is.

Definition

A scoring function $\delta : (\Sigma \cup \{-\})^2 \to \mathbb{Q}$ is called good *if it satisfies the following conditions:*

1. $(\forall a \in (\Sigma \cup \{-\}))(\delta(a,a) = 0)$;
2. $(\forall a,b,c \in (\Sigma \cup \{-\}))(\delta(a,c) \leq \delta(a,b) + \delta(b,c))$ *(triangle inequality).*

This definition immediately leads to:

Lemma

If δ is a good scoring function,

$$\delta(a,b) \geq 0$$

for all $a, b \in (\Sigma \cup \{-\})$.

Proof: $0 = \delta(a,a) \leq \delta(a,b) + \delta(b,a) =^{\star} 2\delta(a,b)$ for all $a, b \in (\Sigma \cup \{-\})$.

$\star$ We required $p(a,b) = p(b,a)$ already for pairwise alignments □

Lemma 5

Let δ a good scoring function, $S = \{c, S_1, \ldots, S_k\}$ a set of words and $T = (V,E)$ the star with center c and leaves $S_1, \ldots, S_k$. Let $(c', S^{(1)}, \ldots, S^{(k)})$ a multiple alignment of S compatible with T. Then for all $i, j \in \{1, \ldots, k\}$

$$\delta(S^{(i)}, S^{(j)}) \leq \delta(S^{(i)}, c') + \delta(c', S^{(j)}) = sim(S_i, c) + sim(c, S_j)$$

holds.

Proof: As the score $\delta(S^{(i)}, S^{(j)})$ of the pairwise alignment is the sum of the scores of the pairs in all columns, the inequality follows immediately from the triangle inequality continued on δ.
The equality follows from the fact that the pairwise alignments of S_i and c resp. c and S_j induced by the multiple alignment $(c', S^{(i)}, \ldots, S^{(k)})$ are optimal and $\delta(-,-) = 0$. $\square$

Theorem

Let δ a good scoring function, δ_{SP} the SP score induced by δ and $\mathcal{S} = \{S_1, \ldots, S_k\}$ a set of words. Furthermore let $sim_{SP}(\mathcal{S})$ the SP score of an optimal multiple alignment for $\mathcal{S}$. Then the multiple alignment $(S^{(1)}, \ldots, S^{(k)})$ constructed by the center star method satisfies

$$\delta_{SP}(S^{(1)}, \ldots, S^{(k)}) \leq \left(2 - \frac{2}{k}\right) \cdot \underline{sim}_{SP}(S_1, \ldots, S_k).$$

Proof: Let $(\bar{S}^{(1)}, \ldots, \bar{S}^{(k)})$ an optimal multiple alignment of $\mathcal{S}$ wrt. SP scoring i.e. $\delta_{SP}(\bar{S}^{(1)}, \ldots, \bar{S}^{(k)}) = sim_{SP}(S_1, \ldots, S_k)$. We define

$$v(S^{(1)}, \ldots, S^{(k)}) := \sum_{1 \leq i \leq k} \sum_{1 \leq j \leq k} \delta(S^{(i)}, S^{(j)}) = 2 \cdot \delta_{SP}(S^{(1)}, \ldots, S^{(k)})$$

and

$$v(\bar{S}^{(1)}, \ldots, \bar{S}^{(k)}) := \sum_{1 \leq i \leq k} \sum_{1 \leq j \leq k} \delta(\bar{S}^{(i)}, \bar{S}^{(j)}) = 2 \cdot \delta_{SP}(\bar{S}^{(1)}, \ldots, \bar{S}^{(k)})$$

$$= 2 \cdot sim_{SP}(\mathcal{S}).$$

Then the claim follows from

$$\frac{v(S^{(1)}, \ldots, S^{(k)})}{v(\bar{S}^{(1)}, \ldots, \bar{S}^{(k)})} \leq \left(2 - \frac{2}{k}\right).$$

Let

$$m := \min_{\underline{t}\in\mathcal{S}} \sum_{s\in\mathcal{S}} sim(s,t) = \sum_{s\in\mathcal{S}} sim(c,s) = \sum_{s\in\mathcal{S}\underline{\setminus\{c\}}} sim(c,s).$$

WLOG we assume $c = S_k$. Lemma 5 then assures

$$\begin{aligned} & v(S^{(1)},\dots,S^{(k)}) \\ &= \sum_{1\le i\le k}\sum_{1\le j\le k} \delta(S^{(i)},S^{(j)}) \le \sum_{\substack{1\le i\le k\\1\le j\le k}} (sim(S_i,c)+sim(S_j,c)) \\ &= \sum_{1\le i\le k-1}\sum_{1\le j\le k-1} (sim(S_i,c)+sim(S_j,c)) \\ &= \sum_{1\le i\le k-1}\sum_{1\le j\le k-1} sim(S_i,c) + \sum_{1\le i\le k-1}\sum_{1\le j\le k-1} sim(S_j,c) \\ &= 2\cdot(k-1)\cdot \sum_{1\le i\le k-1} sim(S_i,c) = 2\cdot(k-1)\cdot m. \end{aligned}$$

On the other hand

$$\begin{aligned} v(\bar{S}^{(1)},\dots,\bar{S}^{(k)}) &= \sum_{1\le i\le k}\sum_{1\le j\le k} \delta(\bar{S}^{(i)},\bar{S}^{(j)}) \ge \sum_{1\le i\le k}\sum_{1\le j\le k} sim(S_i,S_j) \\ &\ge k\cdot \sum_{1\le j\le k} sim(c,S_j) = k\cdot m, \end{aligned}$$

as each choice of i results in

$$\sum_{1\le j\le k} sim(S_i,S_j) \ge \sum_{1\le j\le k} sim(c,S_j)$$

since $c := \operatorname{argmin}_{t\in\mathcal{S}} \sum_{s\in\mathcal{S}} sim(t,s)$. This leads us to

$$\frac{v(S^{(1)},\dots,S^{(k)})}{v(\bar{S}^{(1)},\dots,\bar{S}^{(k)})} \le \frac{2\cdot(k-1)\cdot m}{k\cdot m} = 2-\frac{2}{k}.$$

$\square$

We have shown that for good scoring functions and SP scoring the center star method is an $\left(2-\frac{2}{k}\right)$-approximation algorithm for the multiple alignment of k words.
The center star method is, however, not only suitable for SP scoring. One can show that when scoring based on consensus words it is a 2-approximation if using good scoring functions.

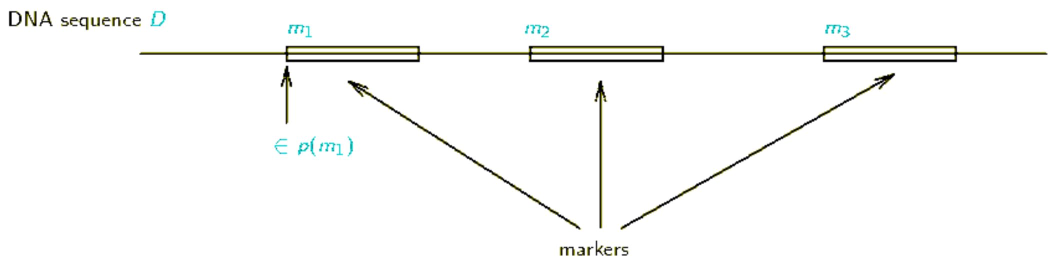
Sequencing

Physical Mapping (first approach for sequencing): Break down many copies of the DNA molecule in question in many overlapping fragments.
$\Rightarrow$ Order in the original molecule is lost.

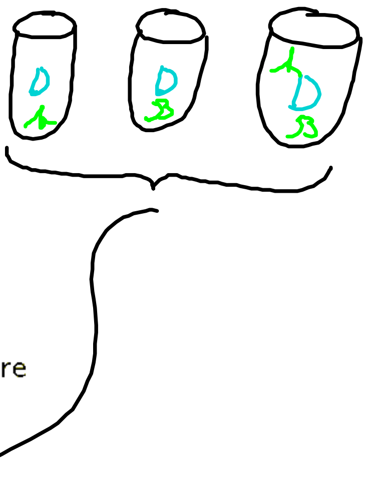
Target: Determine a *physical map* describing the placement of the fragments. This is done using *markers*, being short given base sequences.

Definition

Let D a DNA sequence. A physical map *for D consists of a set M of markers and a function $p : M \to 2^{\mathbb{N}}$, giving all occurrences of m in D (via positions) for each marker $m \in M$. Creating a physical map is called* mapping.



Restriction site mapping: Use restriction enzymes to create the fragments. The restriction sites are used as markers.
Method: Double digest method
Input: A DNA molecule D to be mapped and two restriction enzymes $\mathcal{A}$ and $\mathcal{B}$ with disjoint restriction sites.
Steps:



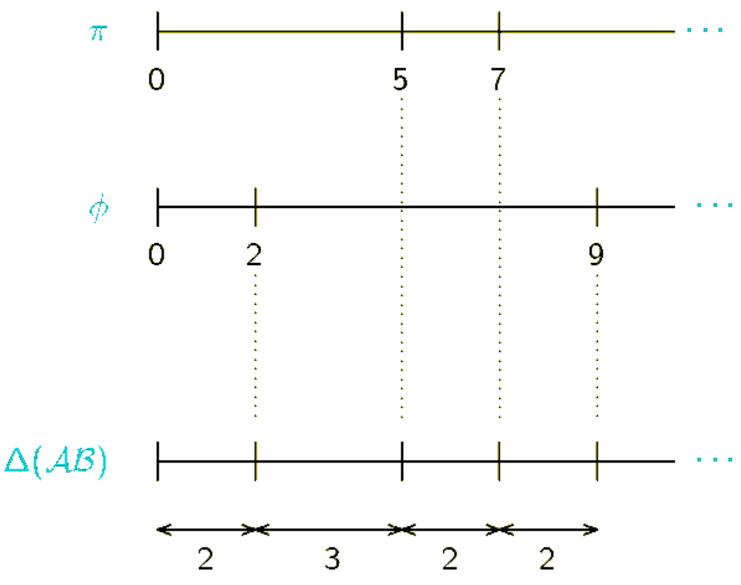
1. Create three copies of D.
2. In three different test-tubes use enzyme $\mathcal{A}$, enzyme $\mathcal{B}$ and enzymes $\mathcal{A}$ and $\mathcal{B}$ resp. on one copy of D each. Now each test-tube contains an unordered set of fragments of D.
3. Determine length of these fragments to create multisets (Here $[x, i]$ denotes x appearing i times)

   $\Delta(X)$, $X \in \{\mathcal{A}, \mathcal{B}, \mathcal{AB}\}$, where
   $\Delta(X) = \{[\ell, i] \in \mathbb{N}^2 \mid \text{digestion of } D \text{ by enzyme(s) } X \text{ created } i \text{ fragments of length } \ell\}$.

We consider the idealized case of complete digestion.

Goal: Derive arrangement of fragments from sets $\Delta(X)$.

To do this we look for arrangements π and ϕ of the fragments in multisets $\Delta(\mathcal{A})$ and $\Delta(\mathcal{B})$. These arrangements should be such that the positions where fragments end induce just the fragment lengths in $\Delta(\mathcal{AB})$.



Definition

Let $X := \{[x_1, i_1], \ldots, [x_n, i_n]\}$ *be a multiset over* $\mathbb{N}$*. Let* $\pi = (x_{j_1}, \ldots, x_{j_l})$, $l := \sum_{1 \leq k \leq n} i_k$, *be a permutation of the elements of* X*. We define*

$$Pos(\pi) := \{0, x_{j_1}, x_{j_1} + x_{j_2}, \ldots, \sum_{1 \leq k \leq l} x_{j_k}\}$$

as the position set of permutation π*. For* $Y = \{y_1, \ldots, y_l\}$, $y_i \in \mathbb{N}_0$, $0 \leq i \leq l$, *and* $y_1 < y_2 < \cdots < y_l$ *we define the multiset*

$$Dist(Y) := \{[\ell, k] \mid |\{i \mid |y_{i+1} - y_i| = \ell \wedge i \in [1 : l-1]\}| = k \in \mathbb{N}\}.$$

We call $Dist(Y)$ distance set of Y.

Note: $Dist(Pos(\pi)) = X$ for each permutation π of the elements in X.

Definition

Let A, B *and* C *multisets over* $\mathbb{N}$*. Furthermore let* π *(resp.* ϕ*) an arrangement of the elements of* A *(resp.* B*). The pair* (π, ϕ) *is called* feasible for A, B and C, *if*

$$Dist(Pos(\pi) \cup Pos(\phi)) = C.$$

Double digest problem (DDP): Given multisets A, B and C. Determine a feasible pair of arrangements of the elements in A and B.

Requirement: The elements of the three sets have the same sum.

Brute force: Test all permutations of A and B.
$\Rightarrow$ In the worst case (each element occurs only once in A resp. B) $(|A|)! \cdot (|B|)!$ many alternatives.

Theorem

Dec-DDP is NP-complete

Proof: Problem is in NP, as it is obviously possible to test in polynomial time, if a pair (π, ϕ) of permutations is a feasible solution.

Completeness: Reduction from set partition to Dec-DDP.

Set partition problem: Input: Set X of naturals. Is there a partition of X into sets Y and Z whose elements have the same sum?

Details: See exercises.

Further problem: The solution for a given input is not unique. If e.g. (π, ϕ) is a feasible solution, (π^r, ϕ^r) is also feasible, where x^r denotes the reverse order of x. There are also instances with additional solutions.

Method: Partial digest method
Input: A DNA molecule D to be mapped and a restriction enzyme $\mathcal{A}$.
Steps:

1. Generate multiple copies of D.
2. Use enzyme $\mathcal{A}$ on D in several experiments with different durations. Each results in a set of fragments.
3. Determine the length of these fragments and collect all the lengths in a multiset $\Delta_p(\mathcal{A})$.

Idealised model of data:

Definition

Let D a DNA molecule and $c_1, \dots, c_n$ the possible restriction sites of restriction enzyme $\mathcal{A}$ in D, $c_1 < c_2 < \cdots < c_n$ and $c_0 = 0$, $c_{n+1} = |D|$. We call set $\Delta_p(\mathcal{A})$ determined with the partial digest method ideal, *if*

$$\Delta_p(\mathcal{A}) = \\ \{[\underline{\ell}, \underline{k}] \mid |\{(i,j) \in \mathbb{N}^2 \mid c_j - c_i = \ell \wedge 0 \leq i < j \leq n+1\}| = k \in \mathbb{N}\}$$

holds, i.e. if the length of each possible fragment is counted exactly once in $\Delta_p(\mathcal{A})$.

Can we determine the arrangement of the fragments from an ideal set $\Delta_p(\mathcal{A})$ efficiently?

We abstract $\Delta_p(\mathcal{A})$ as a multiset of $\binom{n}{2}$ elements, where $n-2$ is the number of restriction sites of $\mathcal{A}$ in D.

Definition

Let A be a multiset of $\binom{n}{2}$ elements from $\mathbb{N}$ and let $P = \{x_1, \dots, x_n\}$ be a set of elements from $\mathbb{N}_0$ where $0 = x_1 < x_2 < \cdots < x_n$. We call such a set P a point set *and define the multiset*

$$Dist_p(P) := \\ \{[\underline{\ell}, \underline{k}] \mid |\{(i,j) \in \mathbb{N}^2 \mid \underbrace{x_j - x_i}_{} = \ell \wedge 1 \leq i < j \leq n\}| = k \in \mathbb{N}\}$$

of all pairwise distances of points in P. The point set P is called feasible solution for A, *if*

$$Dist_p(P) = A$$

holds.

Partial digest problem (PDP): Given a multiset A with $\binom{n}{2}$ elements from $\mathbb{N}$ determine a point set P that is a feasible solution for A, or 0, if there is no such P.

Intuitively the partial digest problem thus is to reconstruct the position set P of restriction sites from the multiset A of fragment lengths.

Difference to DDP: Because of partial digestion overlaps of fragments are possible.

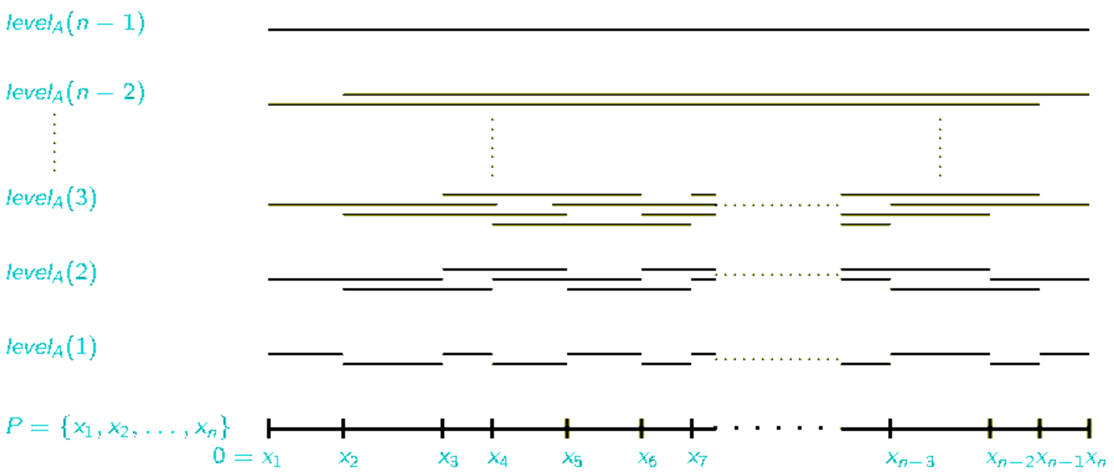
Characterizing feasible solutions:

Definition

Let A be a multiset of $\binom{n}{2}$ elements from $\mathbb{N}$ and let $P = \{x_1, \ldots, x_n\}$, $x_i \in \mathbb{N}_0$, $1 \le i \le n$, $0 = x_1 < x_2 < \cdots < x_n$ a feasible solution for PDP with input $\Delta_p(A)$. For $1 \le i < n$ we define

$$level_A(i) := \{[\ell, k] \mid |\{j \in \mathbb{N} \mid \underbrace{x_{j+i} - x_j} = \ell \wedge 1 \le j \le n - i\}| = k\} \subseteq A$$

as the multiset of distances of end points in P having an index distance of i.



Let A be a multiset of $\binom{n}{2}$ elements from $\mathbb{N}$ and let $P = \{x_1, x_2, \ldots, x_n\}$ be a feasible solution for PDP with input A. We make the following observations:

- $|level_A(i)| := n - i$.
- In our biological motivation $level_A(1)$ is the multiset of fragment lengths whose fragments are restricted by neighbored restriction sites (or ends of the DNA molecule). Thus we call the elements of $level_A(1)$ as *atomic distances*.
- $level_A(n-1) = \{y_{max}\}$ where $y_{max} := \max(A)$. Thus $level_A(n-1)$ only contains the length of the DNA sequence.
- Considering the union of multisets

  $$level_A(1) \,\dot\cup\, level_A(2) \,\dot\cup \cdots \dot\cup\, level_A(n-1) = A,$$

  holds.

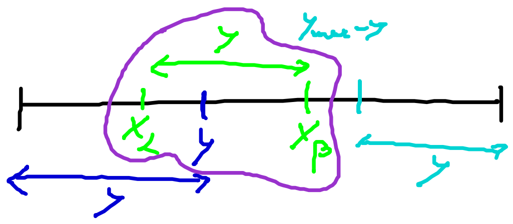
Brute force: Consider all of the maximal $\binom{\binom{n}{2}}{n-1}$ possibilities of choosing atomic distances and check their $(n-1)!$ arrangements for feasibility.

Notation: We define

$$\delta(X, \underset{\in \text{ Punkt}}{\underline{y}}) := \{[\ell, k] \mid |\{x \in X \mid |x - y| = \ell\}| = k \in \mathbb{N}\}$$

for $X = \{x_1, \ldots, x_n\}$ a set of naturals and y a natural.

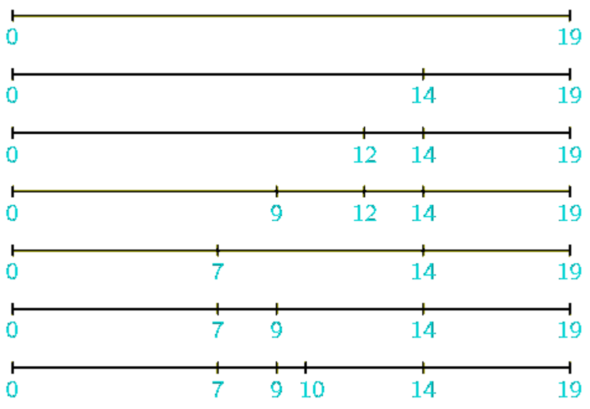
Assumption: An input for PDP has $\binom{n}{2}$, $n \in \mathbb{N}$, many elements as otherwise there is no solution.

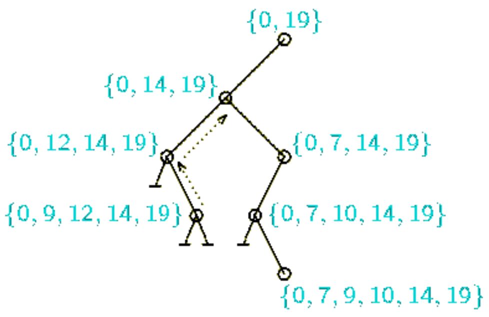


Input: A multiset A of $\binom{n}{2}$ elements from $\mathbb{N}$.
Steps:

1. Sort elements in A by size.
2. $S := \varepsilon$ (empty stack).
3. $y_{max} := \max(A)$; $X := \{0, y_{max}\}$; $A := A \setminus \{y_{max}\}$.
4. Place further elements by calling recursive procedure Plaziere(X,A,S), working as follows: (see handout)

Example: $A = \{[1,1],[2,\underline{1}],[3,1],[4,1],\ [5,\underline{2}],\ [7,\underline{2}],[9,2],[10,2],$
$[\underline{12,1}],\ [\underline{14,1]},\ \underline{[19,1]}\}.$





dead end, no placement at left (resp. right) end possible.

Backtracking, saved state is restored.